From computation to a reconstruction of (linear) logic

Team LoVe – LIPN Université Sorbone Paris Nord

Boris ENG (advisor: Thomas Seiller)

Context

Foundations of logic

Traditional proof theory logic $\longrightarrow$ mathematical tools

Context

Foundations of logic

Traditional proof theory logic ⟶ mathematical tools

Transcendental Syntax (Jean-Yves Girard) mathematical tools ⟶ logic (emergence)

Context

Foundations of logic

Traditional proof theory logic ⟶ mathematical tools

Transcendental Syntax (Jean-Yves Girard) mathematical tools ⟶ logic (emergence)

↳ from an interactive model of computation (think of a society)

Context

Foundations of logic

Traditional proof theory logic $\longrightarrow$ mathematical tools

Transcendental Syntax (Jean-Yves Girard) mathematical tools $\longrightarrow$ logic (emergence)

↳ from an interactive model of computation (think of a society)

↳ behaviours : interaction $\rightsquigarrow$ classification

Context

Foundations of logic

Traditional proof theory logic $\longrightarrow$ mathematical tools

Transcendental Syntax (Jean-Yves Girard) mathematical tools $\longrightarrow$ logic (emergence)

↳ from an interactive model of computation (think of a society)

↳ behaviours : interaction $\rightsquigarrow$ classification

↳ types : pre-made tests $\rightsquigarrow$ classification

Context

Foundations of logic

Traditional proof theory logic $\longrightarrow$ mathematical tools

Transcendental Syntax (Jean-Yves Girard) mathematical tools $\longrightarrow$ logic (emergence)

↳ from an interactive model of computation (think of a society)

↳ behaviours : interaction $\rightsquigarrow$ classification

↳ types : pre-made tests $\rightsquigarrow$ classification

My thesis : turn it into a technical work.

Context

Foundations of logic

Traditional proof theory logic $\longrightarrow$ mathematical tools

Transcendental Syntax (Jean-Yves Girard) mathematical tools $\longrightarrow$ logic (emergence)

↳ from an interactive model of computation (think of a society)

↳ behaviours : interaction $\rightsquigarrow$ classification

↳ types : pre-made tests $\rightsquigarrow$ classification

My thesis : turn it into a technical work.

↳ Assumption : a reconstruction of logic starts from linear logic.

Context

Foundations of logic

Traditional proof theory logic $\longrightarrow$ mathematical tools

Transcendental Syntax (Jean-Yves Girard) mathematical tools $\longrightarrow$ logic (emergence)

↳ from an interactive model of computation (think of a society)

↳ behaviours : interaction $\rightsquigarrow$ classification

↳ types : pre-made tests $\rightsquigarrow$ classification

My thesis : turn it into a technical work.

↳ Assumption : a reconstruction of logic starts from linear logic.

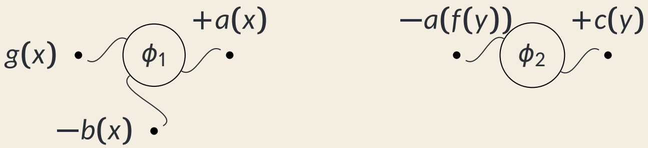
↳ Goal : make the logical mechanisms explicit.

Stellar Resolution

The space of computation

Independent stars with (un)polarised first-order term as rays.

Constellations (kind of programs) as multisets of stars.

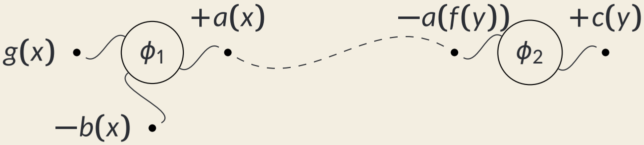


Stellar Resolution

The space of computation

Independent stars with (un)polarised first-order term as rays.

Constellations (kind of programs) as multisets of stars.



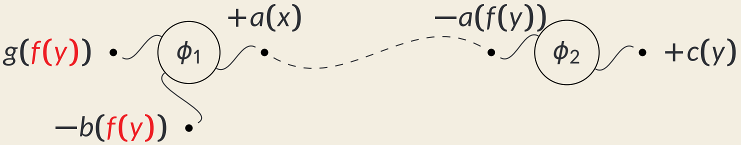
t and u are matchable with unifier $\theta = \{x \mapsto f(y)\}$.

Stellar Resolution

The space of computation

Independent stars with (un)polarised first-order term as rays.

Constellations (kind of programs) as multisets of stars.



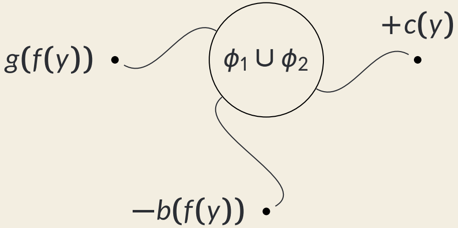
t and u are matchable with unifier $\theta = \{x \mapsto f(y)\}$.

Stellar Resolution

The space of computation

Independent stars with (un)polarised first-order term as rays.

Constellations (kind of programs) as multisets of stars.



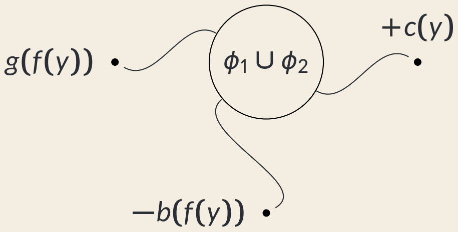
t and u are matchable with unifier $\theta = \{x \mapsto f(y)\}$.

Stellar Resolution

The space of computation

Independent stars with (un)polarised first-order term as rays.

Constellations (kind of programs) as multisets of stars.



t and u are matchable with unifier $\theta = \{x \mapsto f(y)\}$.

Accidentally : (query-free) logic programming and tiling meet (e.g DNA computing).

What is a proof ?

From proof trees to proof structures

Proof tree π **:**

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash B, B^{\perp}}\,\mathsf{ax} \qquad \dfrac{}{\vdash A, A^{\perp}}\,\mathsf{ax}}{\vdash A^{\perp}, B^{\perp}, B \otimes A}\,\otimes}{\vdash A^{\perp} \parr B^{\perp}, B \otimes A}\,\parr}{\vdash (A^{\perp} \parr B^{\perp}) \parr (B \otimes A)}\,\parr
$$

What is a proof ?

From proof trees to proof structures

Proof tree π :

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash B, B^{\perp}}\,\mathsf{ax} \qquad \dfrac{}{\vdash A, A^{\perp}}\,\mathsf{ax}}{\vdash A^{\perp}, B^{\perp}, B \otimes A}\,\otimes}{\vdash A^{\perp} \parr B^{\perp}, B \otimes A}\,\parr}{\vdash (A^{\perp} \parr B^{\perp}) \parr (B \otimes A)}\,\parr$$

Linear Logic proof structure $\mathscr{S}$ (more general) :

$$\begin{tikzcd}
A^{\perp} \arrow[dr] \arrow[rrrrrr, bend left, leftrightarrow] & & B^{\perp} \arrow[dl] \arrow[rr, bend left, leftrightarrow] & & B \arrow[dr] & & A \arrow[dl] \\
& A^{\perp} \parr B^{\perp} \arrow[drr] & & & & B \otimes A \arrow[dll] & \\
& & & C := (A^{\perp} \parr B^{\perp}) \parr (B \otimes A) & & &
\end{tikzcd}$$

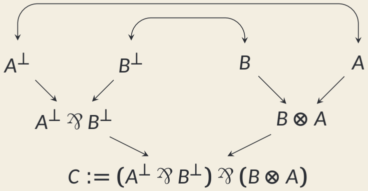
What is a proof ?

From proof trees to proof structures

Proof tree π :

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash B, B^{\perp}}\,\mathsf{ax} \qquad \dfrac{}{\vdash A, A^{\perp}}\,\mathsf{ax}}{\vdash A^{\perp}, B^{\perp}, B \otimes A}\,\otimes}{\vdash A^{\perp} \parr B^{\perp}, B \otimes A}\,\parr}{\vdash (A^{\perp} \parr B^{\perp}) \parr (B \otimes A)}\,\parr$$

Linear Logic proof structure $\mathscr{S}$ (more general) :



Logical correctness : does $\mathscr{S}$ pass tests $T_1, \ldots, T_n$? If so, proof of C.

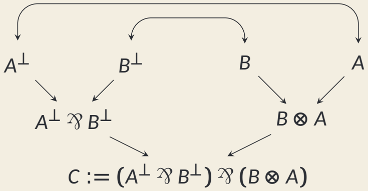
What is a proof ?

From proof trees to proof structures

Proof tree π :

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash B, B^\perp}\,\mathrm{ax} \qquad \dfrac{}{\vdash A, A^\perp}\,\mathrm{ax}}{\vdash A^\perp, B^\perp, B \otimes A}\,\otimes}{\vdash A^\perp \parr B^\perp, B \otimes A}\,\parr}{\vdash (A^\perp \parr B^\perp) \parr (B \otimes A)}\,\parr$$

Linear Logic proof structure $\mathscr{S}$ (more general) :



Logical correctness : does $\mathscr{S}$ pass tests $T_1, \ldots, T_n$? If so, proof of C.

Translation into constellations : correct structure = core constellation + set of tests

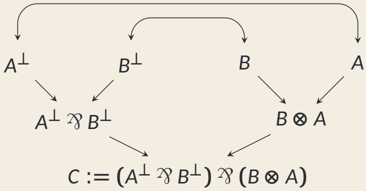
What is a proof ?

From proof trees to proof structures

Proof tree π :

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash B, B^{\perp}}\,\mathrm{ax} \qquad \dfrac{}{\vdash A, A^{\perp}}\,\mathrm{ax}}{\vdash A^{\perp}, B^{\perp}, B \otimes A}\,\otimes}{\vdash A^{\perp} \parr B^{\perp}, B \otimes A}\,\parr}{\vdash (A^{\perp} \parr B^{\perp}) \parr (B \otimes A)}\,\parr$$

Linear Logic proof structure $\mathscr{S}$ (more general) :



Logical correctness : does $\mathscr{S}$ pass tests $T_1, \ldots, T_n$? If so, proof of C.

Translation into constellations : correct structure = core constellation + set of tests

↳ typing by stereotypes : passing $T_1, \ldots, T_n$ implies $\Phi : C$.

Behaviours

using realisability techniques

Behaviours

using realisability techniques

Typing by behaviour : classify from how Φ interacts.

Behaviours

using realisability techniques

Typing by behaviour : classify from how Φ interacts.

Pre-behaviour set of constellations (programs) **A**.

Behaviours

using realisability techniques

Typing by behaviour : classify from how Φ interacts.

Pre-behaviour set of constellations (programs) **A**.

Orthogonality Define "good interaction".

Behaviours

using realisability techniques

Typing by behaviour : classify from how Φ interacts.

Pre-behaviour set of constellations (programs) $\mathbf{A}$.

Orthogonality Define "good interaction".

↳ for instance $\Phi \perp \Phi' \iff \mathrm{Ex}(\Phi \uplus \Phi')$ terminates.

Behaviours

using realisability techniques

Typing by behaviour : classify from how Φ interacts.

Pre-behaviour set of constellations (programs) $\mathbf{A}$.

Orthogonality Define "good interaction".

↳ for instance $\Phi \perp \Phi' \iff \mathrm{Ex}(\Phi \uplus \Phi')$ terminates.

↳ $\mathbf{A}^{\perp}$ set of good partners.

Behaviours

using realisability techniques

Typing by behaviour : classify from how Φ interacts.

Pre-behaviour set of constellations (programs) $\mathbf{A}$.

Orthogonality Define "good interaction".

↳ for instance $\Phi \perp \Phi' \iff \mathrm{Ex}(\Phi \uplus \Phi')$ terminates.

↳ $\mathbf{A}^{\perp}$ set of good partners.

Behaviour when $\mathbf{A} = \mathbf{A}^{\perp\perp}$.

Behaviours

using realisability techniques

Typing by behaviour : classify from how Φ interacts.

Pre-behaviour set of constellations (programs) $\mathbf{A}$.

Orthogonality Define "good interaction".

↳ for instance $\Phi \perp \Phi' \iff \mathrm{Ex}(\Phi \uplus \Phi')$ terminates.

↳ $\mathbf{A}^{\perp}$ set of good partners.

Behaviour when $\mathbf{A} = \mathbf{A}^{\perp\perp}$.

Tensor $\mathbf{A} \otimes \mathbf{B} := \{\Phi_A \uplus \Phi_B \mid \Phi_A \in \mathbf{A}, \Phi_B \in \mathbf{B}\}^{\perp\perp}$.

Behaviours

using realisability techniques

Typing by behaviour : classify from how Φ interacts.

Pre-behaviour set of constellations (programs) **A**.

Orthogonality Define "good interaction".

↳ for instance $\Phi \perp \Phi' \iff \mathrm{Ex}(\Phi \uplus \Phi')$ terminates.

↳ $\mathbf{A}^{\perp}$ set of good partners.

Behaviour when $\mathbf{A} = \mathbf{A}^{\perp\perp}$.

Tensor $\mathbf{A} \otimes \mathbf{B} := \{\Phi_A \uplus \Phi_B \mid \Phi_A \in \mathbf{A}, \Phi_B \in \mathbf{B}\}^{\perp\perp}$.

Other "connectives" $\mathbf{A} \parr \mathbf{B} := \mathbf{A}^{\perp} \otimes \mathbf{B}^{\perp}$ and $\mathbf{A} \multimap \mathbf{B} := \mathbf{A}^{\perp} \parr \mathbf{B}$.

Behaviours

using realisability techniques

Typing by behaviour : classify from how Φ interacts.

Pre-behaviour set of constellations (programs) $\mathbf{A}$.

Orthogonality Define "good interaction".

↳ for instance $\Phi \perp \Phi' \iff \mathrm{Ex}(\Phi \uplus \Phi')$ terminates.

↳ $\mathbf{A}^{\perp}$ set of good partners.

Behaviour when $\mathbf{A} = \mathbf{A}^{\perp\perp}$.

Tensor $\mathbf{A} \otimes \mathbf{B} := \{\Phi_A \uplus \Phi_B \mid \Phi_A \in \mathbf{A}, \Phi_B \in \mathbf{B}\}^{\perp\perp}$.

Other "connectives" $\mathbf{A} \parr \mathbf{B} := \mathbf{A}^{\perp} \otimes \mathbf{B}^{\perp}$ and $\mathbf{A} \multimap \mathbf{B} := \mathbf{A}^{\perp} \parr \mathbf{B}$.

Adequation $\Phi \in \mathbf{A}$ behaves as expected from the tests for $\mathbf{A}$.

Conclusion and future works

A lot of ways to extend the idea.

Conclusion and future works

A lot of ways to extend the idea.

- extension to full linear logic, second and first order.

Conclusion and future works

A lot of ways to extend the idea.

- extension to full linear logic, second and first order.

  ↳ better design for logic ?

Conclusion and future works

A lot of ways to extend the idea.

- extension to full linear logic, second and first order.
  ↳ better design for logic ?
- hopes in complexity theory (descriptive ?).

Conclusion and future works

A lot of ways to extend the idea.

- extension to full linear logic, second and first order.
  ↳ better design for logic ?
- hopes in complexity theory (descriptive ?).
  ↳ better understanding of logic, better understanding of complexity ?

Conclusion and future works

A lot of ways to extend the idea.

- extension to full linear logic, second and first order.
  ↳ better design for logic ?
- hopes in complexity theory (descriptive ?).
  ↳ better understanding of logic, better understanding of complexity ?

Thank you for listening !